Proceedings of the 2001 IEEE/RSJ
International Conference on Intelligent Robots and Systems
Maui, Hawaii, USA, Oct. 29 – Nov. 03, 2001

Linguistic Control of Mobile Robots

Magnus Egerstedt*

magnuse@hrl.harvard.edu
Div. of Engineering and Applied Sciences
Harvard University
Cambridge, MA 02138, USA

Abstract

In this paper we study the interactions between symbolic computer programs and mechanical devises, e.g. mobile robots. We show that by using motion description languages for generating continuous motions from symbolic input strings, this interaction between the continuous and the discrete can be given a meaningful control theoretic interpretation. We furthermore illustrate how robot behaviors can be learned within this framework. We also investigate how to choose the motion description languages in order to minimize the lengths of the inputs to the robots.

1 Introduction

The use of computers for controlling mechanical devises, such as robots or machine tools, has brought to the fore the need for a systematic study of the interactions between the symbolic computer programs and the continuous devise dynamics. Typically, this interaction is defined by the way the continuous machine operates on the outputs from a computer generated *motion control program*. Such a motion control program generates strings of symbolic inputs to a real time system that controls the devise based on sensory information about internal states as well as the environment it is operating in. In order to understand the interactions between these two heterogeneous components, different *hybrid architectures*, serving as abstractions between continuous and discrete control, have been suggested. In [4] a general model for such hybrid systems is proposed: $\dot{x} = f(x, y, v(\lfloor p \rfloor))$, $\dot{p} = g(x, y, v(\lfloor p \rfloor))$, $y = h(x, v(\lfloor p \rfloor))$, where x is the continuous state of the system and y is the measured continuous output signal. Moreover, v is the symbolic input string from the motor control program and the evolution of the scalar p triggers the reading of that string. Here $\lfloor \cdot \rfloor$ denotes the floor operator, and g is assumed to be nonnegative for all arguments.

In this paper we model the way linguistic control signals affect mechanical devises on this form. We will show how to choose the symbolic instruction set in such a way that it is rich enough to support the generation of continuous control commands that make robots carry out navigation tasks in cluttered environments. We will also construct an interpreter mechanism for generating these control commands from the symbolic inputs.

The symbols in the *motion alphabet* that we use for generating the continuous motions should represent different control actions that, when applied to a specific machine, define particular motion segments. This idea has been made concrete by using *motion description languages* (MDLs), as suggested in [3, 4, 8]. A MDL is a language given by a set of symbolic strings that represent idealized motions. Different authors have used different types of letters in the motion alphabet, but here we let them be triples (u, k, ξ), where u is an open-loop component, k is a feedback mapping, and ξ is an interrupt function that tells the system when a new input symbol, or control triple, should be read.

2 Kinematic Machines and Motion Description Languages

The primary objects of study in this paper are so called *motion description languages* (MDLs). Given a finite set, or alphabet, A, by A^* we understand the

*Supported in part by the US Army Research Office, Grant number DAAG 5597-1-0114, and in part by the Sweden-America Foundation 2000 Research Grant.

0-7803-6612-3/01/$10.00©2001 IEEE

set of all strings of finite length over A. There is a naturally defined binary operation on this set, namely the concatenation of strings, denoted by $a_1 \cdot a_2$, i.e. $a_1 \cdot a_2 \in A^\star$ if $a_1, a_2 \in A^\star$. Relative to this operation, $A^\star$ is a semigroup. If we include the empty string in $A^\star$ it becomes a monoid, i.e. a semigroup with an identity, and a formal language is a subset of a free monoid over a finite alphabet.

Now, by a *motion alphabet* we mean a possibly infinite set of symbols representing different control actions that, when applied to a specific machine, define segments of motion. A MDL is thus given by a set of symbolic strings that represent idealized motions, i.e. a MDL is a subset of a free monoid over a given motion alphabet. Particular choices of MDLs become meaningful only when the language is defined relative to the physical devise, or *kinematic machine*, that is to be controlled. By a kinematic machine we understand the tuple $M = (U, X, Y, F, G, H)$, where U is an input space, X is a state space, Y is an output space, $F : X \to TX$ and $G : X \to TX$ are vector fields, and $H : X \to Y$ is an output map. The evolution of the machine is given by $\dot{x} = F(x) + G(x)u,\ y = H(x)$.

Each letter in the motion alphabet corresponds to a control command that generates a particular motion segment on a given kinematic machine. We let the letters in the motion alphabet that are read by M be triples of the form (u, k, ξ), where $u : \mathbb{R} \to U, k : Y \to U$, and $\xi : \mathbb{R} \times Y \to \{0,1\}$. If, at time t_0, M receives the input string $(u_1, k_1, \xi_1), \ldots, (u_p, k_p, \xi_q)$, then x evolves according to

$$\begin{aligned} \dot{x} &= F(x) + G(x)(u_1 + k_1(y)); \quad && t_0 \le t < T_1 \\ &\vdots && \vdots \\ \dot{x} &= F(x) + G(x)(u_p + k_p(y)); && T_{q-1} \le t < T_q, \end{aligned}$$

where T_i denotes the time at which the interrupt ξ_i changes from 0 to 1.

The model of a trigger based hybrid system as described in the introduction can moreover reproduce this behavior if symbols are interpreted and operated on as follows. Let the input string be such that $v(i) = (u_i, k_i, \xi_i),\ i \in \mathbb{Z}^+$, and let

$$\begin{aligned} \dot{x} &= f(x, y, v(\lfloor p \rfloor)) = f(x, y, (u_{\lfloor p \rfloor}, k_{\lfloor p \rfloor}, \xi_{\lfloor p \rfloor})) \\ &= F(x) + G(x)(u_{\lfloor p \rfloor} + k_{\lfloor p \rfloor}(y)) \\ y &= h(x, v(\lfloor p \rfloor)) = h(x, (u_{\lfloor p \rfloor}, k_{\lfloor p \rfloor}, \xi_{\lfloor p \rfloor})) \\ &= H(x) \\ \dot{p} &= g(x, y, v(\lfloor p \rfloor)) = g(x, y, (u_{\lfloor p \rfloor}, k_{\lfloor p \rfloor}, \xi_{\lfloor p \rfloor})) \\ &= \begin{cases} 0 & \text{if } \xi_{\lfloor p \rfloor}(t, y) = 0 \\ \delta_t & \text{if } \xi_{\lfloor p \rfloor}(t, y) = 1, \end{cases} \end{aligned}$$

where δ_t is a unit impulse at time t, $x(0) = x_0$, and $p(0) = 1$.

It is clear that we now have a construction that allows continuous machines to operate on linguistic inputs in a way that can be given a meaningful control theoretic interpretation.

2.1 Reinforcement Learning

One of the advantages of letting the symbolic inputs correspond directly to control commands is that it is possible to let robots learn continuous-time behaviors in a computationally feasible way.

Consider a discrete time system whose states evolve on $X \subset \mathbb{R}^n$. The space of admissible controls is given by $U \subset \mathbb{R}^m$, and transitions are generated according to $x((k+1)\Delta) = \lambda(x(k\Delta), u(k\Delta))$, where Δ is the sample time of the system. Given an initial state $x(0) = x_0$ and a control policy $\pi : X \to U$, a discounted, cumulative cost can be defined as

$$V^\pi(x_0) = \sum_{k=0}^{\infty} \gamma^k r\big(x(k\Delta), \pi(x(k\Delta))\big),$$

where $\gamma \in (0,1)$, and $r : X \times U \to \mathbb{R}^+$ is the incremental cost. (See for example [9].)

Now, the Bellman equation associated with the problem of minimizing V over all possible policies is

$$V^\star(x) = \min_{u \in U} \big\{ r(x,u) + \gamma V^\star(\lambda(x,u)) \big\},\ \forall x \in X,$$

where $V^\star$ denotes the optimal value function. The Bellman equation directly gives that the optimal policy satisfies

$$\pi^\star(x) \in \arg\min_{u \in U} \big\{ r(x,u) + \gamma V^\star(\lambda(x,u)) \big\},\ \forall x \in X.$$

These types of discrete-time *reinforcement learning* problems have successfully been implemented and solved using the *Q-learning* method. Proof of convergence has been given for finite-state Markovian decision tasks [13] and for linear discrete-time systems with infinite state and control spaces (see for example [2]), which is the case in our discrete-time dynamical system example.

Using a Q-function, the discrete-time learning problem can be formulated in this setting as

$$\begin{aligned} Q^\star(x,u) &= r(x,u) \\ &\quad + \gamma \min_{v \in U} Q^\star(\lambda(x,u), v),\ \forall x \in X, u \in U \\ \pi^\star(x) &\in \arg\min_{u \in U} Q^\star(x,u),\ \forall x \in X. \end{aligned}$$

The advantage with the Q-learning approach is that it enables us to iteratively approximate the Q-function, and in [1] this was done in a provenly convergent way using stochastic approximation techniques. The idea is, at the pth iteration where the learner is given the data pair (x_p, u_p), to let the Q-function be updated as

$$\begin{aligned} Q_{p+1}(x,u) &= Q_p(x,u) - \alpha_p g_p(x,x_p,u,u_p) \cdot \\ &\cdot \Big(Q_p(x_p,u_p) - r(x_p,u_p) - \\ &- \gamma \min_{v \in U} Q_p(\lambda(x_p,u_p),v)\Big), \end{aligned}$$

$\forall x \in X, u \in U$, where $\alpha_p = \alpha_0 p^{-1}$, $\alpha_0 > 0$, and $\{g_p\}$ is a sequence of Hilbert space kernel functions [1].

However, if the system is evolving in continuous time, the optimal value function must satisfy the continuous time Hamilton-Jacobi-Bellman equation (see for example [5]), which constitutes a notoriously hard problem. But, by letting the system learn letters from a motion alphabet, we can treat the continuous problem as a discrete-time problem. What the robot needs to learn is simply what triples (u, k, ξ) to use, i.e. if we assume that we observe the states of the system, we want the robot to learn a policy $\pi : X \to U \times (X \to U) \times (\mathbb{R} \times X \to \{0,1\})$. A direct modification of the previously stated discrete-time, stochastic approximation algorithm can be made to achieve this.

2.2 Instruction Complexity

One of the driving motivations behind this work was to understand what role feedback plays if one wants to minimize the number of symbolic instructions that one needs to send to a given robot. In other words, is it possible to reduce the length of the words over a given motion alphabet by allowing feedback, compared to the purely open-loop case? In order to capture the difference in the complexity of operating a robot with and without reference to sensory information, i.e. when using closed and open-loop instructions, we have to define a complexity measure that captures this difference in a meaningful way.

Definition 2.1 *(Instruction Complexity) Given a kinematic machine M and a motion description language $\mathcal{L}$, we say that the instruction complexity for realizing a given task T is given by the minimum length word $u = (u_1, k_1, \xi_1), \ldots, (u_p, k_p, \xi_p) \in \mathcal{L}$ that guarantees a successful execution of T on M. We denote this by $C_T(\mathcal{L}, M)$.*

3 Mobile Robots

3.1 Basic Definitions

What makes the control of mobile robots particularly challenging is the fact that the robots operate in unknown, or partially unknown environments. Any attempt to model such a system must take this into account, and we choose to do this by letting the robot observe certain facts about the environment. We let the robot be defined as follows:

Definition 3.1 *(M_1) Let the kinematic machine M_1 be given by $\dot{x} = v$, $x, v \in \mathbb{R}^2$, $y_1 = x$, $y_2 = c_f(x)$, $y_1, y_2 \in \mathbb{R}^2$, where c_f is the contact force from the environment.*

Remark 3.1 *The contact force from an obstacle could either be generated by tactile sensors in contact with the obstacle or by range sensors such as sonars, lasers, or IR-sensors.*

Relative to this machine it is now possible to define the following two MDLs for distinguishing between the open-loop and the closed-loop cases.

Definition 3.2 *(Open-Loop and Closed-Loop MDLs) Let the Open-Loop MDL, $\mathcal{L}_{ol}$, be given by the free monoid over the set*

$$\{(u,k,\xi) \mid u \in \mathbb{R}^2,\ k = 0, \qquad \xi : \mathbb{R} \to \{0,1\}\}.$$

Consequently, let the Closed-Loop MDL, $\mathcal{L}_{cl}$, be given by the free monoid over the set

$$\{(u,k,\xi) \mid u = 0,\ k(y) \in \{\kappa(x_F - y_1), Dy_2\}, \\ \xi : \mathbb{R}^2 \to \{0,1\}\},$$

where $\kappa > 0$ is a constant, D is a linear map from $\mathbb{R}^2$ to $\mathbb{R}^2$, and x_F is the final, desired robot position.

By a *point-to-point navigation task* we understand the problem of moving the robot between given initial and final states in a safe way. (The robot should not intersect the interior of any obstacle.) We denote this task $P2P$. The problem that we try to solve is thus: *Find the two instruction complexities $C_{P2P}(\mathcal{L}_{ol}, M_1)$ and $C_{P2P}(\mathcal{L}_{cl}, M_1)$.* The reason why we believe that M_1 is an appropriate machine, and $\mathcal{L}_{ol}$ and $\mathcal{L}_{cl}$ are appropriate MDLs is that they are simple enough to allow us to compute bounds on the complexities $C_{P2P}(\mathcal{L}_{ol}, M_1)$ and $C_{P2P}(\mathcal{L}_{cl}, M_1)$. At the same time they are expressive enough for solving the point-to-point navigation task, as we will see in the paragraphs

to follow. The paths generated by $\mathcal{L}_{ol}$ on M_1 are furthermore identical to those paths that are considered in the literature on the complexity of minimum time or shortest path algorithms for robot motion planning in dynamic environments [6, 10, 12].

3.2 Complexity Theorems

Let the environment $\mathcal{E}$ be given by a compact and convex polygon in $\mathbb{R}^2$, and let N disjunct, compact, and convex polygons with M vertices each be populating the interior of $\mathcal{E}$. For a given obstacle P, $int(P)$ denotes its interior, ∂P its boundary, and $vert(P)$ and $face(P)$ are the sets of vertices and faces in P respectively. We furthermore let $cone(x, P)$ be the smallest, closed convex cone that originates from $x \notin int(P)$ and contains P. We also let $line(x_1, x_2)$ denote the set $\{z \in \mathbb{R}^2 \mid z = \alpha x_1 + (1-\alpha)x_2,\ \alpha \in [0,1]\}$, and we say that $x_2 \in \mathcal{E}$ is *visible* from $x_1 \in \mathcal{E}$ if $line(x_1, x_2) \cap int(P_i) = \emptyset$, $i = 1, \ldots, N$.

Definition 3.3 (Visibility Chain) *$\mathcal{V}(x_0, x_F)$ is said to be a visibility chain if it satisfies the following: $\mathcal{V}(x_0, x_F) = \emptyset$ if x_F is visible from x_0. Otherwise $\mathcal{V}(x_0, x_F) = \{P_1, \ldots, P_k\}$, where P_{j+1} is the obstacle closest to x_F that is visible from P_j. Here P_k is visible from x_F and P_1 is the obstacle closest to x_F that is visible from x_0.*

3.2.1 Open-Loop Control

Theorem 3.1 (Open-Loop Complexity) *In a convex environment populated by N convex, polygonal obstacles with $M \geq 3$ vertices each, $C_{P2P}(\mathcal{L}_{ol}, M_1)$ is of order $\mathcal{O}(NM)$.*

Before we can prove this theorem, the following lemmas are necessary.

Lemma 3.1 *Given two obstacles P and Q, then $\exists p \in vert(P)$ such that $cone(p, Q) \cap P = \{p\}$.*

The proof of this lemma can be found in any textbook on convex analysis. (See for example [11].)

Lemma 3.2 *Given a visibility chain $\mathcal{V}(x_0, x_F)$. It is possible to connect all neighboring obstacles in $\mathcal{V}(x_0, x_F)$ with lines between vertices. Furthermore, a line connecting P_i with P_{i+1} does not intersect any other obstacles in $\mathcal{V}(x_0, x_F)$.*

Proof of Lemma 3.2: Given $P_i, P_{i+1} \in \mathcal{V}(x_0, x_F)$. The existence of a vertex $p_i \in vert(P_i)$ such that $cone(p_i, P_{i+1}) \cap P_i = \{p_i\}$ follows from Lemma 3.1. Thus two vertices in $vert(P_{i+1})$ can be connected by lines from p_i. That neither of these lines intersect any other obstacles in $\mathcal{V}(x_0, x_F)$ follows directly from the definition of the visibility chain. The lemma thus follows. ∎

Proof of the Open-Loop Complexity Theorem: The proof consists of establishing tight bounds on the number of segments necessary for producing a piecewise linear path between x_0 and x_F. This path should furthermore not intersect the interior of any obstacle. To find these bounds is equivalent to finding $C_{P2P}(\mathcal{L}_{ol}, M_1)$ since the only paths that can be generated on M_1, using words in $\mathcal{L}_{ol}$, are piecewise linear.

Construct a visibility chain from x_0 to x_F, and assume that there are N obstacles in $\mathcal{V}(x_0, x_F)$. The proof in this case is inductive.

Let $N = 2$. Since $cone(x_0, P_1)$ contains at least two vertices p_1^1, p_1^2 in P_1, these two vertices can be reached from x_0 using only one linear segment. By virtue of Lemma 3.1, there exists another vertex $p_1 \in vert(P_1)$ such that $cone(p_1, P_2) \cap P_1 = \{p_1\}$. From p_1 two vertices, p_2^1, $p_2^2 \in vert(P_2)$ can thus be reached using only one linear segment. It is furthermore possible to reach p_1 from one of p_1^1 or p_1^2 using at most $\lfloor (M-1)/2 \rfloor$ segments, where $\lfloor \cdot \rfloor$ denotes the floor operator.

Now, from x_F two vertices in P_2 can be reached using one linear segment, and one of these vertices can be connected to one of p_2^1 or p_2^2 using at most $\lfloor M/2 \rfloor - 1$ segments. Thus the total number of linear segments necessary is at most $1 + \lfloor (M-1)/2 \rfloor + \lfloor M/2 \rfloor + 1$.

Next, assume that the bound holds for the case when we have k obstacles, and let $N = k+1$. Pick any obstacle P_i in the interior of the visibility chain and form the path connecting the k obstacles in $\{P_1, \ldots, P_{i-1}, P_{i+1}, \ldots, P_{k+1}\}$ with $(k-1)(1 + \lfloor (M-1)/2 \rfloor) + \lfloor M/2 \rfloor + 1$ segments. From a vertex $p_{i-1} \in vert(P_{i-1})$ construct the two supporting hyperplanes to P_i as in Lemma 3.1, and make a similar construction from a vertex $p_i \in vert(P_i)$ to P_{i+1}. By following these hyperplanes from P_{i-1} to P_i and P_{i+1}, it is straightforward to see that we add a total of $1 + \lfloor (M-1)/2 \rfloor$ segments to our path, that thus consist of $k(1 + \lfloor (M-1)/2 \rfloor) + \lfloor M/2 \rfloor + 1$ segments. That no other obstacles intersect these new line-segments follows from Lemma 3.2.

If our assumption about the visibility chain is untrue, we can assume that k obstacles are not in the chain. Construct the path that connects the remaining $N-k$ obstacles with x_0 and x_F, using $(N-k-1)(1 + \lfloor (M-1)/2 \rfloor) + \lfloor M/2 \rfloor + 1$ segments. The remaining k obstacles can at most intersect k of the segments on the path. Assume that one of the

obstacles, Q, intersects the segment between P_i, P_{i+1}, and let p_i, p_{i+1} be two points on these obstacles that are visible from each other. If we now change our path so that it connects to these two points instead of the previous vertices on P_i, P_{i+1} that were connected, we have, in the worst case, to add a total of 2 segments to our upper bound. Thus our total number of segments are at most $(N-k-1)(1+\lfloor(M-1)/2\rfloor)+\lfloor M/2\rfloor+1+2k$, which is less than or equal to $(N-1)(1+\lfloor(M-1)/2\rfloor)+\lfloor M/2\rfloor+1$ as long as $M \geq 3$.

That this bound is tight can easily be seen by simply constructing an environment where it is tight. The theorem thus follows. ∎

3.2.2 Closed-Loop Control

We now construct a closed-loop control strategy that requires a lower number of instructions than what was necessary in the open-loop case. However, we will not compute $C_{P2P}(\mathcal{L}_{cl}, M_1)$ exactly. Instead we produce upper bounds for $C_{P2P}(\mathcal{L}_{cl}, M_1)$ that are low enough to guarantee that $C_{P2P}(\mathcal{L}_{cl}, M_1) < C_{P2P}(\mathcal{L}_{ol}, M_1)$.

Before we can compute this bound, some comments about how M_1 interacts with the environment must be made. The contact force from an obstacle, P, in contact with the robot is parallel to the outward normal of the surface of the obstacle. However, since no unique normal vector exists when $x \in vert(P)$, we let the output, y_2, take on any value in the *normal cone*, i.e. $y_2 \in N_P(x) = \{h \in \mathbb{R}^2 \mid \langle h, y-x\rangle \leq 0,\ \forall y \in P\}$.

Theorem 3.2 (Closed-Loop Complexity) *In a convex world populated by N convex, polygonal obstacles, an upper bound on $C_{P2P}(\mathcal{L}_{cl}, M_1)$ is of order $\mathcal{O}(N)$.*

The proof of Theorem 3.2 is constructive, and before we can state the proof some preliminary results about how to construct the appropriate control sequence must be established. When the robot is not in contact with an obstacle, we choose to use $k(y) = \kappa(x_F - y_1)$, where $\kappa > 0$, as in Definition 3.2.

When the robot is in contact with an obstacle it seems reasonable to follow the contour of that obstacle, as suggested in [7]. The control strategy that we propose for this guarantees that the robot reaches the unique global minimum (the point closest to x_F on the obstacle), while committing to a clockwise or counter-clockwise obstacle negotiation, before it leaves the contour of the obstacle.

Lemma 3.3 *Given a convex obstacle P. A point $p \in \partial P$ is visible from x_F if and only if $\exists n \in N_P(p)$ such that $\langle n, x_F - p\rangle \geq 0$.*

Proof of Lemma 3.3: That $p \in \partial P$ is not visible from x_F is equivalent to $line(p, x_F) \cap int(P) \neq \emptyset$, which in turn is equivalent to $x_F - p \in int(T_P(p))$, where $T_P(p)$ is the *tangent cone* $T_P(p) = \{h \in \mathbb{R}^2 \mid h = \lambda(y-p), y \in P, \lambda \geq 0\}$.

Now, $T_P(p)$ is also the *polar cone* to $N_P(p)$, i.e. $T_P(p) = \{s \in \mathbb{R}^2 \mid \langle s, y\rangle \leq 0, \forall y \in N_P(p)\}$. Thus $x_F - p \in int(T_P(p))$ is equivalent to $\langle x_F - p, y\rangle < 0, \forall y \in N_P(p)$. This gives that p is visible from x_F if and only if $\exists n \in N_P(p)$ such that $\langle n, x_F - p\rangle \geq 0$, and the lemma follows. ∎

Lemma 3.4 *Given a compact, convex obstacle P, and a point $x \notin P$. Then there exists uniquely a point $p^\star \in P$ that is the closest point to x in P. Furthermore $p^\star$ uniquely satisfies $x - p^\star \in N_P(p^\star)$.*

The proof of Lemma 3.4 follows directly from standard results in convex analysis. (See for example [11].)

It is clear that if the robot moves counter-clockwise along the contour of a convex obstacle, P, then on the part of ∂P that is visible from x_F (in the absence of other obstacles) it holds that $\angle(x_F - y_1, y_2) < 0$ before $p^\star$ is encountered, and $\angle(x_F - y_1, y_2) > 0$ after $p^\star$ is encountered. Here $p^\star$ is the point closest to x_F on ∂P and $\angle(x_F - y_1, y_2) \in (-\pi, \pi]$ is the angle between $x_F - y_1$ and y_2, where y_1 and y_2 are the position of M_1 and the contact force felt by M_1 respectively. (See Definition 3.1.)

By using these results we can now propose the following closed-loop control strategy.

Definition 3.4 (Closed-Loop Control) *The closed-loop controller consists of sequences $(u_1, k_1, \xi_1), (u_2, k_2, \xi_2), (u_3, k_3, \xi_3), \ldots$, where, for $j = 1, 2, \ldots$, the control commands are given by $u_{2j-1} = 0$, $k_{2j-1}(y) = \kappa(x_F - y_1)$, and $\xi_{2j-1}(y, t) = 0$ if $\langle y_2, x_F - y_1\rangle \geq 0$ and equal to 1 otherwise. Furthermore, $u_{2j} = 0$, $k_{2j}(y) = cR(-\pi/2)y_2$, and $\xi_{2j}(y, t) = 0$ if either $\langle y_2, x_F - y_1\rangle < 0$ or $\angle(x_F - y_1, y_2) < 0$ and equal to 1 otherwise. Here $R(\theta)$ is a 2×2 rotation matrix, and $c > 0$.*

Proof of the Closed-Loop Control Theorem: The proof follows from applying the control sequence in Definition 3.4 to M_1. The robot thus moves along $line(x_0, x_F)$ until an obstacle P is encountered. The robot then follows the contour of P until $\langle y_2, x_F - y_1\rangle \geq 0$ and $\angle(x_F - y_1, y_2) \geq 0$, i.e. when the point on

∂P closest to x_F is encountered. Since the robot always moves closer and closer to x_F, and it has already been at the point in P closest to x_F, it never encounters that obstacle again. So, by not encountering any obstacle twice, we immediately get an upper bound of $2N + 1$ segments, and the theorem follows. ∎

3.2.3 Discussion

We have now derived results that explain how many instructions are necessary when instructing a robot to navigate in cluttered environments populated by polygonal obstacles. As seen in the previous paragraphs, such a question can be formulated and solved quite elegantly when programming robots using motion description languages, and the difference in operation between feedback and open-loop can be captured by restricting the MDLs to have a certain structure.

By summarizing the contributions from Theorems 3.1 and 3.2, the main complexity theorem in this paper is as follows:

Theorem 3.3 *Let the environment be populated by N disjoint, convex, polygonal obstacles with M vertices each. Then $C_{P2P}(\mathcal{L}_{ol}, M_1)$ is of order $\mathcal{O}(NM)$, while an upper bound on $C_{P2P}(\mathcal{L}_{cl}, M_1)$ is of order $\mathcal{O}(N)$.*

What this result means is that when the sensory information available to us is sufficiently abundant, fewer instructions are necessary in the feedback case than in the open-loop case. This way of investigating the length of the input sequence has implications for many areas of robotics. For teleoperated robots, it is clear that a control procedure that requires few instructions is to prefer since the communication channels may be noisy and unreliable. This type of argument also has implications for the way mobile robots should be programmed. The actual controller is typically running at a high frequency, while commands from the motion control program are sent at a much lower frequency. To use a control procedure that requires few input symbols is thus preferable since it frees up computational and communication resources to be used elsewhere in the system.

References

[1] W.L. Baker. *Learnig Via Stochastic Approximation in Function Space.* Doctoral Dissertation, Division of Engineering and Applied Sciences, Harvard University, Cambridge, 1997.

[2] S.J. Bradtke, B.E. Ydstie, and A.G. Barto. Adaptive Linear Quadratic Control Using Policy Iteration. In *American Control Conference*, pp. 3475–3479, 1994.

[3] R.W. Brockett. On the Computer Control of Movement. In the *Proceedings of the 1988 IEEE Conference on Robotics and Automation*, pp. 534–540, New York, April 1988.

[4] R.W. Brockett. Hybrid Models for Motion Control Systems. In *Perspectives in Control*, Eds. H. Trentelman and J.C. Willems, pp. 29–54, Birkhäuser, Boston, 1993.

[5] A.E. Bryson and Y.C. Ho. *Applied Optimal Control: Optimization, Estimation, and Control.* Hemisphere Publishing Corporation, New York, 1975.

[6] J. Canny. *The Complexity of Robot Motion Planning.* The MIT Press, Cambridge, MA, 1987.

[7] J.E. Hopcroft and G. Wilfong. Motion of Objects in Contact. *The International Journal of Robotics Research*, Vol. 4, No. 4, pp. 32–46, 1986.

[8] V. Manikonda, P.S. Krishnaprasad, and J. Hendler. Languages, Behaviors, Hybrid Architectures and Motion Control. In *Mathematical Control Theory*, Eds. Willems and Baillieul, pp. 199–226, Springer-Verlag, 1998.

[9] T.M. Mitchell. *Machine Learning.* McGraw-Hill Series in Computer Science, WCB/McGraw-Hill, Boston, 1997.

[10] J. Reif and M. Sharir. Motion Planning in the Presence of Moving Obstacles. *IEEE 26th Symposium on Foundations of Computer Science*, pp. 144-154, 1985.

[11] R.T. Rockafellar. *Convex Analysis.* Princeton University Press, Princeton, NJ, 1970.

[12] M. Sharir and A. Schorr. On Shortest Paths in Polyhedral Spaces. *SIAM Journal of Computer Science*, Vol. 15, No. 1, pp. 192–215, 1986.

[13] C.J.C.H. Watkins and P. Dayan. Q-Learning. *Machine Learning*, Vol. 8, No. 3/4, pp. 257–277, May 1992.